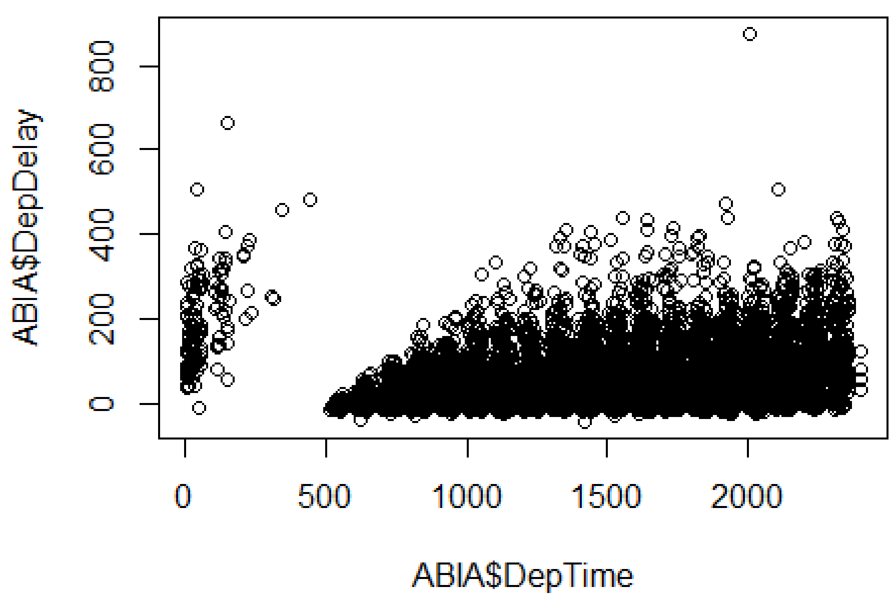
STA 380 Homework 2

Liuxuan Yu

August 14, 2016

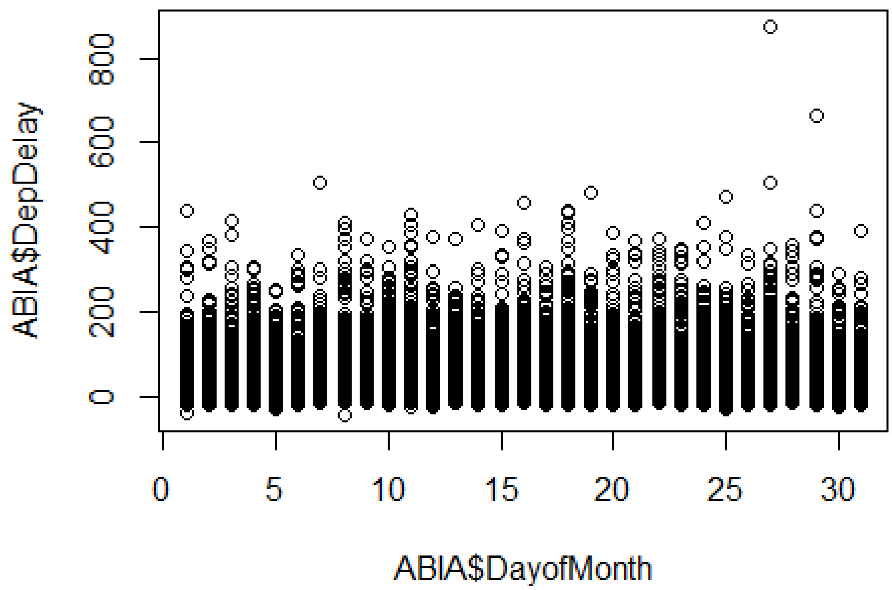
1.Flights at ABIA

```
library(dplyr)

library(ggplot2)

library(cowplot)

library(Hmisc)

setwd("C:\\Users\\Administrator\\Desktop\\R script in class")
ABIA<-read.csv('ABIA.csv',header=T,sep=',')
attach(ABIA)

plot(ABIA$DepTime,ABIA$DepDelay)
```

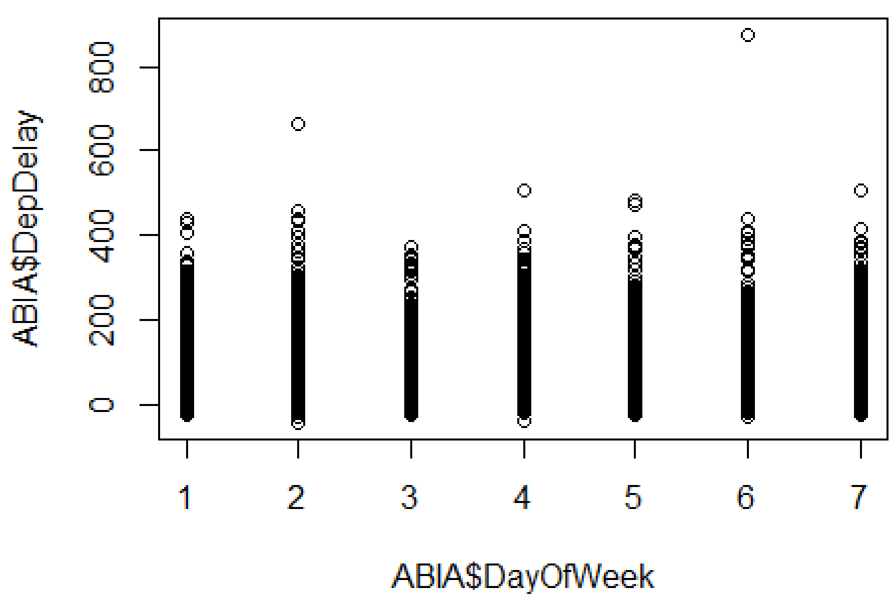


Take plane at morning will decrease the randomness of being delayed

```
plot(ABIA$DayofMonth,ABIA$DepDelay)
```

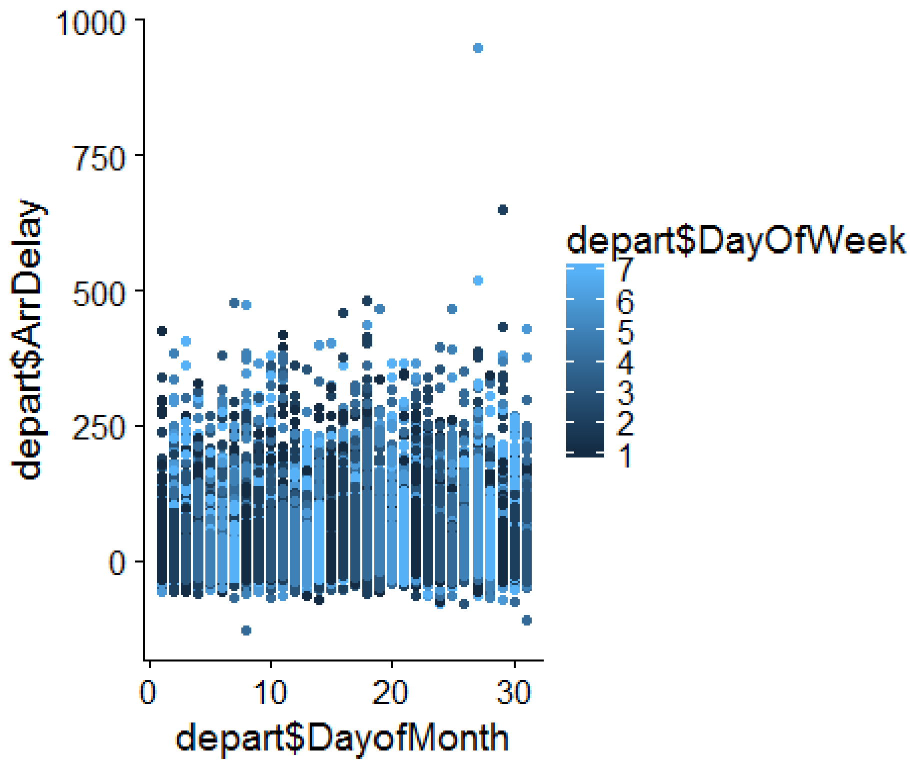



```
plot(ABIA$DayOfWeek,ABIA$DepDelay)
```

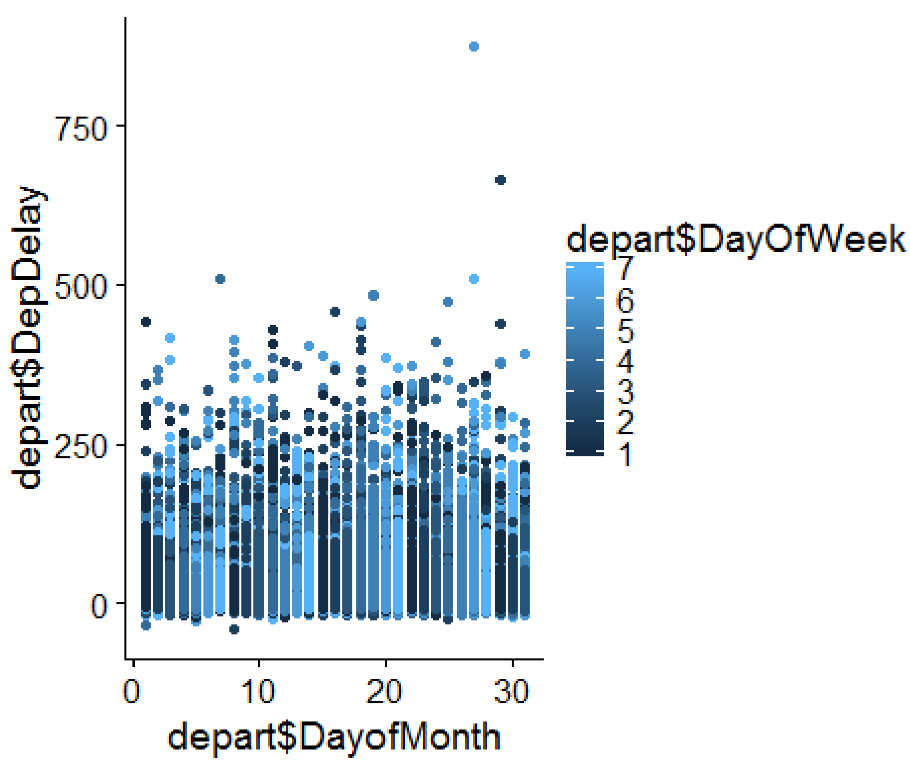



```
depart=ABIA[,c(1:22)]
depart<-depart[complete.cases(depart),]
```

```r
qplot(depart$DayofMonth, depart$ArrDelay,color=depart$DayOfWeek)
```

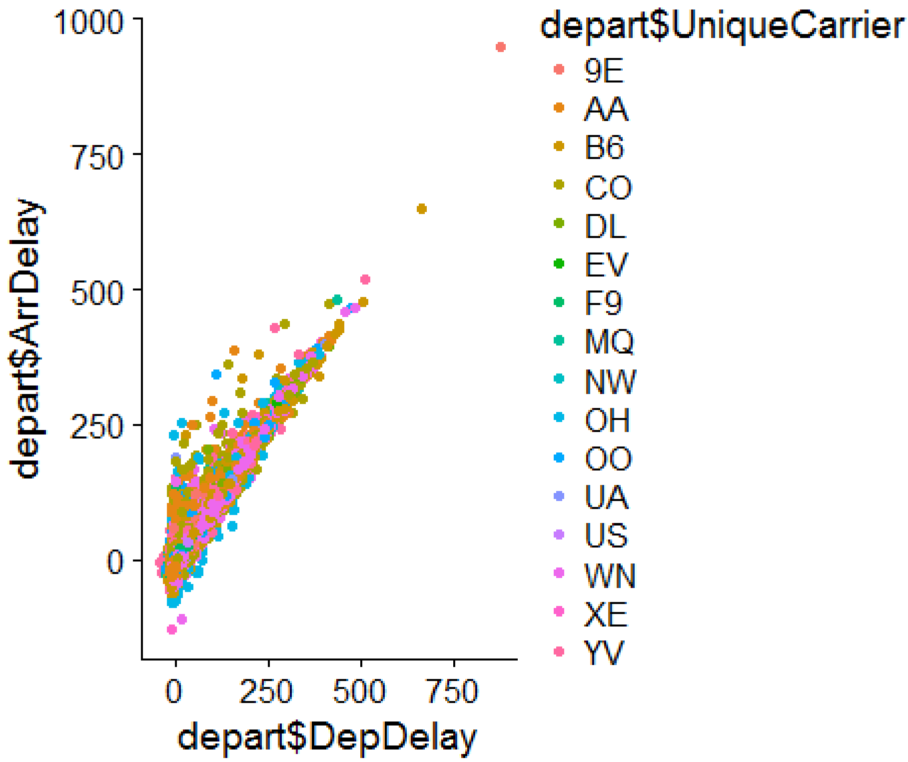



```r
qplot(depart$DayofMonth, depart$DepDelay,color=depart$DayOfWeek)
```

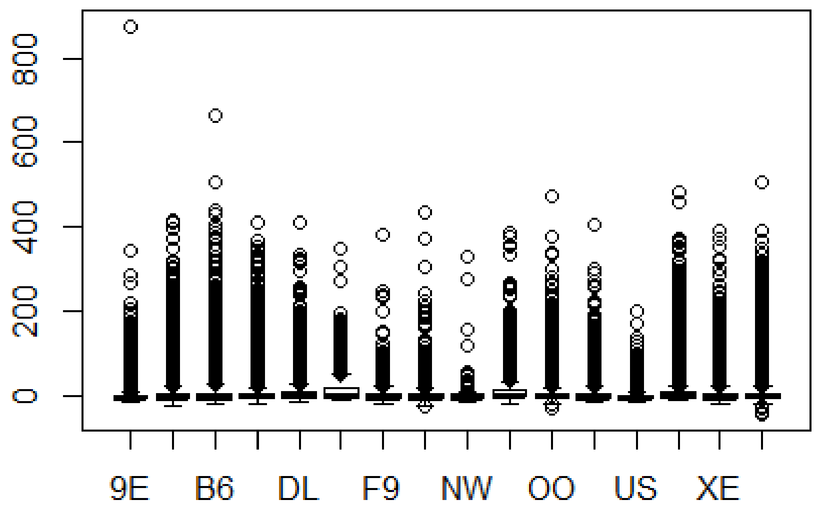


It seems that day of month won't influnece a lot on the delay time.

```
#look at the delay of different flight company
qplot(depart$DepDelay,depart$ArrDelay,color=depart$UniqueCarrier)
```

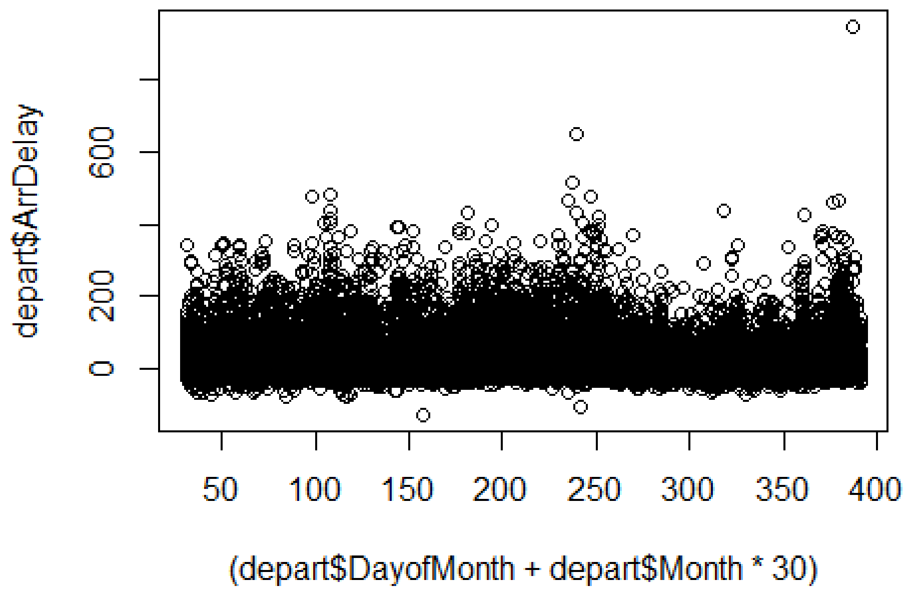



```
describe(depart$UniqueCarrier)

## depart$UniqueCarrier
##        n missing  unique
##    97659       0      16
##
##             9E    AA   B6   CO   DL  EV   F9   MQ  NW   OH   OO   UA
US
## Frequency 2488 19401 4726 9103 2109 808 2129 2491 118 2911 3944 1848 14
55
## %            3    20    5    9    2   1    2    3   0    3    4    2
 1
##              WN   XE   YV
## Frequency 34633 4582 4913
## %            35    5    5

plot(depart$UniqueCarrier,depart$DepDelay)
```



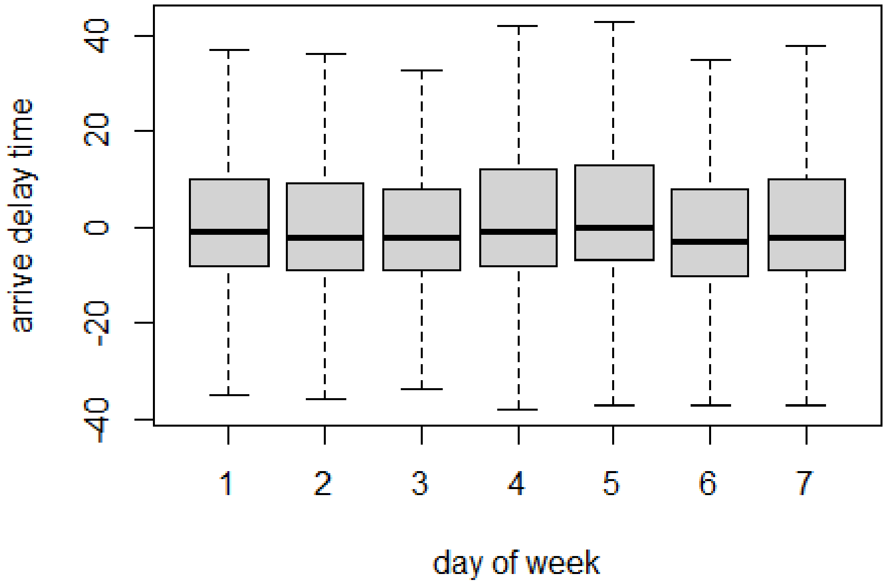
AA and WN contains about half of all the flights in Austin.but AA's delay was not that big as for example B6.

```
#what is the best time of the year to minimize delay
plot((depart$DayofMonth+depart$Month*30),depart$ArrDelay)
```

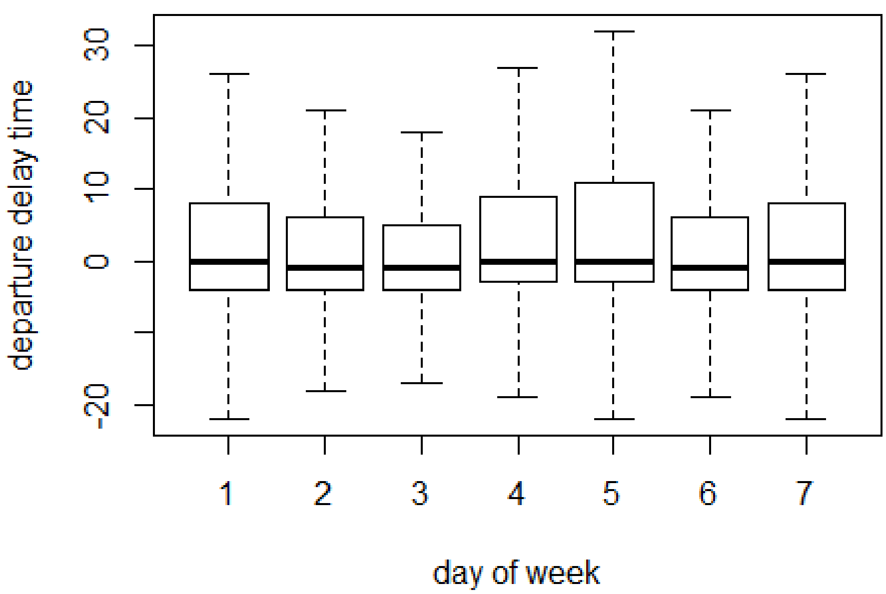


It looks like the days about 260-300 would be better, which is the days about October,and the days in December are very easy to delay, which may have a relationship with Christmas.

```
describe(DayOfWeek)

## DayOfWeek
##        n missing  unique    Info    Mean
##    99260       0       7    0.98   3.902
##
##                1     2     3     4     5     6     7
## Frequency 14798 14803 14841 14774 14768 11454 13822
## %            15    15    15    15    15    12    14

boxplot(ABIA$ArrDelay~ABIA$DayOfWeek,outline=FALSE,xlab='day of week',ylab
='arrive delay time',col = "lightgray")
```




```r
boxplot(ABIA$DepDelay~ABIA$DayOfWeek,outline=FALSE,xlab='day of week',ylab
='departure delay time')
```



With almost same number of data, Friday shows a little bit higher than other weekdays for the arrive delay and departure delay time. Weekends and Wednesday would be lower in both time and time range.

2.Author attribution

```
setwd("C:\\Users\\Administrator\\Desktop\\R script in class")
rm(list=ls())
library(tm)

library(plyr)

readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
            id=fname, language='en') }

## Rolling two directories together into a single corpus
author_dirs = Sys.glob('ReutersC50/C50train/*')
author_dirs = author_dirs[1:50]
file_list = NULL
labels = NULL
for(author in author_dirs) {
  author_name = substring(author, first=21)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
  labels = append(labels, rep(author_name, length(files_to_add)))
}

# Need a more clever regex to get better names here
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

my_corpus = Corpus(VectorSource(all_docs))
names(my_corpus) = file_list

# Preprocessing
my_corpus = tm_map(my_corpus, content_transformer(tolower)) # make everyth
ing lowercase
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers)) # remove
 numbers
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation)) # re
move punctuation
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace)) ## rem
ove excess white-space
my_corpus = tm_map(my_corpus, content_transformer(removeWords), stopwords
("SMART"))

DTM = DocumentTermMatrix(my_corpus)
DTM = removeSparseTerms(DTM, 0.95)
DTM
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 641)>>
## Non-/sparse entries: 180911/1421589
## Sparsity           : 89%
## Maximal term length: 18
## Weighting          : term frequency (tf)

# Now a dense matrix
X_train = as.matrix(DTM)
```

For test data,just read it in the same way as the train data, but later I will delete the author name in the test data.

```
author_dirs = Sys.glob('ReutersC50/C50test/*')
author_dirs = author_dirs[1:50]
file_list = NULL
labels_test = NULL
author_test =NULL
for(author in author_dirs) {
  author_name = substring(author, first=20)
  author_test = append(author_test,author_name)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
  labels_test = append(labels_test, rep(author_name, length(files_to_add)))
}

# Need a more clever regex to get better names here
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

my_corpus = Corpus(VectorSource(all_docs))
names(my_corpus) = file_list

# Preprocessing
my_corpus = tm_map(my_corpus, content_transformer(tolower))
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers))
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation))
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace))
my_corpus = tm_map(my_corpus, content_transformer(removeWords), stopwords
("SMART"))

DTM = DocumentTermMatrix(my_corpus)
DTM = removeSparseTerms(DTM, 0.95)
X_test = as.matrix(DTM)

#delete author name in test data
row.names(X_test)<-c(1:2500)

#fill in the different data between train and test data set
bind_matrix=rbind.fill.matrix(X_train,X_test)
bind_matrix[is.na(bind_matrix)] <- 0
train=bind_matrix[1:2500,]
test=bind_matrix[2501:5000,]
```

```
row.names(train)<-row.names(X_train)
train<-train[,order(colnames(train))]
test<-test[,order(colnames(test))]
```

Method 1: calculate the angle between every test and train vectors, choose the smallest angle of every pairs, and use the train name as the test predictor.

```
predict=NULL
for (j in c(1:2500)){
  cat("Validation",j,"of",2500,"\n")
  list=NULL
  for (i in c(2501:5000)){
    a=bind_matrix[i,]
    b=bind_matrix[j,]
    theta <- cos(sum(a*b)/(sqrt(sum(a*a))*sqrt(sum(b*b))))
    list=append(list,theta)
    }
  predict=append(predict,row.names(train)[which.max(list)])
  }

#predict

author_name = substring(predict, first=20)
author_name= regmatches(author_name, regexpr('.+/',author_name))
true_name=regmatches(row.names(train), regexpr('.+/',row.names(train)))
true_name= substring(true_name, first=20)[1:10]
summary(true_name==author_name)

##    Mode   FALSE    TRUE    NA's
## logical    1851     649    0
```

The result shows that within the 2500 test data, there are 649 true prediction, 1851 wrong prediction, which means that the accuracy of this model is about 25.96%

Method 2: Naive Bayes: compare each product of test and train vector to get the maximum log probabilities

```
smooth_count = 1/nrow(X_train)
w_AP = rowsum(X_train + smooth_count,labels)
w_AP= w_AP/sum(w_AP)
w_train = log(w_AP)

DTM_test = DocumentTermMatrix(test_corpus,list(dictionary=colnames(DT
M)))
DTM_test

## <<DocumentTermMatrix (documents: 2500, terms: 3076)>>
## Non-/sparse entries: 311938/7378062
## Sparsity           : 96%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

```r
X_test = as.matrix(DTM_test)

predict = NULL
for (i in 1:50) {

    cat("Validation",i,"of",50,"\n")
    max = -(Inf)
    list = NULL
  for (j in 1:50) {
    alpha = sum(w_train[j,]*X_test[i,])
    if(alpha > max) {
      max = alpha
      author = rownames(w_train)[j]
    }
  }
  predict = append(predict, list)
}

predict_results = table(labels_test,predict)
correct = NULL
for (i in 1:nrow(predict_results)) {
  correct = append(correct, predict_results[i, i])
}
```

Let's take a look at the authors whose articles are most difficult to guess.

```r
pred_correct = data.frame(author_list, correct)
pred_correct <- pred_correct[order(-correct),]
pred_correct $correct_rate <- pred_correct$correct/50
pred_correct
```

```
##          author_list correct correct_rate
## 29   LynnleyBrowning      49         0.98
## 11    FumikoFujisaki      48         0.96
## 16      JimGilchrist      48         0.96
## 36         NickLouth      46         0.92
## 38     PeterHumphrey      45         0.90
## 21        KarlPenhaul      44         0.88
## 33      MatthewBunce      44         0.88
## 22         KeithWeir      42         0.84
## 40        RobinSidel      42         0.84
## 3     AlexanderSmith      41         0.82
## 28    LynneO'Donnell      40         0.80
## 34     MichaelConnor      40         0.80
## 41      RogerFillion      40         0.80
## 1      AaronPressman      38         0.76
## 12    GrahamEarnshaw      38         0.76
## 6        BradDorfman      37         0.74
## 20       JonathanBirt      37         0.74
## 47    TheresePoletti      37         0.74
## 48        TimFarrand      36         0.72
## 30   MarcelMichelson      35         0.70
```

```
## 39        PierreTran      35          0.70
## 24      KevinMorrison     34          0.68
## 25      KirstinRidley     33          0.66
## 42        SamuelPerry     32          0.64
## 19       JohnMastrini     31          0.62
## 26 KouroshKarimkhany     31          0.62
## 27          LydiaZajc     31          0.62
## 43       SarahDavison     31          0.62
## 45        SimonCowell     31          0.62
## 18           JoeOrtiz     30          0.60
## 14         JanLopatka     28          0.56
## 10        EricAuchard     27          0.54
## 37    PatriciaCommins     27          0.54
## 23     KevinDrawbaugh     26          0.52
## 32         MartinWolk     25          0.50
## 2          AlanCrosby     23          0.46
## 5       BernardHickey     22          0.44
## 49         ToddNissen     22          0.44
## 17     JoWinterbottom     20          0.40
## 31       MarkBendeich     20          0.40
## 15      JaneMacartney     19          0.38
## 35         MureDickie     19          0.38
## 13  HeatherScoffield     17          0.34
## 7   DarrenSchuettler     14          0.28
## 4     BenjaminKangLim     12          0.24
## 50       WilliamKazer     12          0.24
## 8         DavidLawder     10          0.20
## 9       EdnaFernandes     10          0.20
## 44        ScottHillis      6          0.12
## 46           TanEeLyn      2          0.04
```

```
sum(pred_correct$correct)/nrow(X_test)

## [1] 0.6028
```

By using this model, the accuracy rate even reach to 60%,which is much better than the first one.

From the result, we can find that, the LynnleyBrowning is the most easy one to predict, the TanEeLyn is the difficult one to predict.

3.Practice with association rule mining

This question is about the association rule mining.Use the data on grocery purchases to find some interesting association rules for these shopping baskets. Reading the grocery purchases data by using "scan",we try to use the lift, confidence and support to explain the correlation between those items.

```
library(arules)

library(plyr)
```

```r
setwd("C:\\Users\\Administrator\\Desktop\\R script in class")
rm(list=ls())
#read the data into R with "scan"
groceries<-scan("groceries.txt",what="character",sep = "\n",quiet=TRUE)

#First split each row of data into a list of lots of stuffs
grocery_split<- strsplit(groceries, ",")

#Remove duplicates ("de-dupe")
grocery_split<- lapply(grocery_split, unique)

#Cast this variable as a special arules "transactions" class.
grocery<- as(grocery_split, "transactions")
```

Then we run the 'apriori' algorithm. The support value of {X} with respect to the total database is the proportion of transactions in the database which contains the item-set {X} The confidence value of {X} to {Y} measures how item Y appears in baskets that contains X. The lift is supp(X&Y) divided by supp(X)*supp(Y)

Let's look at rules with support > .01 & confidence >.5 to find out the frequent itemsets. The support number is very small because in the data set, the biggest support is only 0.02

```r
rules <- apriori(grocery, parameter=list(support=.01, confidence=.5))

# Look at the output
inspect(rules)

##    lhs                        rhs                      support confidence
 lift
## 1  {curd,

##     yogurt}               => {whole milk}       0.01006609  0.5823529 2.2
79125
## 2  {butter,

##     other vegetables}     => {whole milk}       0.01148958  0.5736041 2.2
44885
## 3  {domestic eggs,

##     other vegetables}     => {whole milk}       0.01230300  0.5525114 2.1
62336
## 4  {whipped/sour cream,

##     yogurt}               => {whole milk}       0.01087951  0.5245098 2.0
52747
## 5  {other vegetables,

##     whipped/sour cream}   => {whole milk}       0.01464159  0.5070423 1.9
84385
## 6  {other vegetables,

##     pip fruit}            => {whole milk}       0.01352313  0.5175097 2.0
25351
```

```
## 7  {citrus fruit,

##      root vegetables}    => {other vegetables} 0.01037112  0.5862069 3.0
29608
## 8  {root vegetables,

##      tropical fruit}     => {other vegetables} 0.01230300  0.5845411 3.0
20999
## 9  {root vegetables,

##      tropical fruit}     => {whole milk}       0.01199797  0.5700483 2.2
30969
## 10 {tropical fruit,

##      yogurt}             => {whole milk}       0.01514997  0.5173611 2.0
24770
## 11 {root vegetables,

##      yogurt}             => {other vegetables} 0.01291307  0.5000000 2.5
84078
## 12 {root vegetables,

##      yogurt}             => {whole milk}       0.01453991  0.5629921 2.2
03354
## 13 {rolls/buns,

##      root vegetables}    => {other vegetables} 0.01220132  0.5020921 2.5
94890
## 14 {rolls/buns,

##      root vegetables}    => {whole milk}       0.01270971  0.5230126 2.0
46888
## 15 {other vegetables,

##      yogurt}             => {whole milk}       0.02226741  0.5128806 2.0
07235
```

```r
#Choose a subset
inspect(subset(rules, subset=lift > 2))
```

[omit the output]

We explore some big lift number which lift is > 2, from the result we know that those X and Y are might dependent on one another, which might be useful for predicting the consequent in future data sets. For example, whole milk are always highly correlated with yogurt, other vegetables are also correlated with root vegetables and tropical fruit.

```r
inspect(subset(rules, subset=confidence > 0.5))
```

[omit the output]

The biggest confidence are still below 0.6, but a lot of them are above 0.5. For those with 0.5 confidence, it means that for the transactions that contains X(lhs), only about 50% that they also contain Y(rhs). So the correlation was not that strong.

```
# get a higher support threshold
inspect(subset(rules, subset=support > .015 & confidence > 0.5))
```

```
##    lhs                             rhs          support    confidence
## 10 {tropical fruit,yogurt}      => {whole milk} 0.01514997 0.5173611
## 15 {other vegetables,yogurt} => {whole milk} 0.02226741 0.5128806
##    lift
## 10 2.024770
## 15 2.007235
```

Now the right hand side only have whole milk, and the yogurt shows a lot, which might show that the yogurt has a high correlation with whole milk.